An Ontological Examination of Object Interaction in Conceptual Modeling

August 15, 2001

Abstract

The object paradigm has found its way from the programming and implementation stage of information systems development into conceptual modeling in systems analysis stage. Conceptual modeling is concerned with modeling and understanding real world domains. To analyze conceptual modeling methods we use ontology, the branch of philosophy that deals with what exists or is assumed to exist in the world.

In this paper, we examine certain concepts of the object approach and the object modeling language UML from an ontological perspective. Specifically, we analyze concepts related to object interactions, namely, associations, messages, and methods. Our analysis indicates that associations and messages are implementation related and thus have no direct ontological equivalent. Hence, we claim that these concepts are not well-suited to model interactions in real-world domains. We suggest to use other concepts to model interactions in object-oriented conceptual modeling. Based on the ontological analysis we derive rules to guide the use of object concepts as modeled in UML.

1 Introduction

Conceptual modeling is characterized as the "activity of formally describing some aspects of the physical and social world for the purposes of understanding" [16]. This understanding is the first step in the information system analysis and design (ISAD) process. The result of this step is a conceptual model.

The next step in the ISAD process is system design, resulting in a model of the information system. This model differs from the description of the real world in the analysis phase because the context is increasingly shaped by technical considerations. The use of the object oriented approach and specifically UML for the design phase is well accepted.

However, it is not clear whether the use of object oriented approaches, especially UML, is suitable also for conceptual modeling. This is the subject of this study. We address this question using ontology. Ontology is the branch of philosophy that deals with what exists or is assumed to exist in the real world. A specific ontology is one set of such assumptions.

Previous effort in examining UML from an ontological perspective has focused on static model aspects (especially, on objects, properties, and classes) [11]. Here we examine dynamic system behavior. Mapping object concepts to ontology not only examines the suitability of the object approach and UML specifically for modeling the real world, it may also lead us to rules and guidelines on *how* to use it for that purpose. These rules may not be obvious or applicable when the object approach is used for systems design.

Method of this study For the purposes of this investigation, we chose Bunge's [6] ontological model as adapted to information systems analysis by Wand and Weber [22, 23]. For abbreviation, we refer to it as the BWW Ontology. While other ontological models have been used to analyze information systems concepts, we use this ontology for several reasons. In particular:

- It is well formalized in terms of concepts, definitions, and premises (based on set theory).
- It has been used in several works to analyze information systems modeling methods [e.g. 13, 18, 21, 23], thus providing a benchmark for evaluating other models.

- It has been applied to analyze object-oriented concepts [22].
- It has been used to generate predictions about conceptual modeling that have been empirically corroborated [e.g. 2, 3, 12]

This study follows the notion of *ontological expressiveness* of modeling approaches [23]. The mapping from the constructs of a modeling language into ontological concepts is called *interpretation mapping*. By analyzing this mapping, we can identify language elements that have no ontological counterpart (construct excess) or have multiple ontological interpretations (construct overload). Use of a construct without ontological meaning may lead to an ontologically meaningless model. Construct overload may lead to an ambiguous model.

The inverse mapping, from the ontological constructs into the modeling language constructs, is called *representation mapping* and can identify construct deficits, i.e. the modeling language does not provide a construct to express an ontologically relevant aspect of the real world. This may lead to incomplete models.

The remainder of this paper introduces the central concepts of the BWW-Ontology as related to our inquiry (Sec. 2) and then examines UML in light of these concepts (Sec. 3). The paper concludes with an outline of the need for further research.

2 Concepts of the BWW-Ontology

The world is made up of *things* that are substantial in the sense that they are perceived to physically exist. Things possess *properties* that are either *intrinsic* or *mutual*. Intrinsic properties are ones that a thing possesses by itself, e.g. color, whereas mutual properties exist between two or more things, e.g. attractive force or distance. Mutual properties are either *binding*, arising out of interaction of things, or *non-binding*, properties of non-interacting things. An example of a binding property is 'works for' in the sense that 'a person works for a company'. An example of a non-binding property is 'behind' in the sense that 'thing A is behind thing B'.

Things can combine to form composite things that exhibit emergent properties, properties not possessed by any of the components itself.

Properties can be related by *laws* that specify the lawful combinations of properties. Laws can be described in terms of *precedence* of properties: Property B precedes property A iff whenever a thing possesses A, it possesses B.

Properties may or may not be directly observable. Humans create models of things by assigning attributes to things. Thus, properties are known to humans in terms of attributes. For example, "color" is an attribute, representing a physical phenomenon (likely not even known to the observer). Attributes are modeled in terms of state functions which are functions on time. Similar things can be modeled via a *functional schema* - a common set of attribute functions. The set of values of the state functions at a given time comprises the *state* of the thing.

A change of state is termed an *event*. A state may be *stable* - that is, will not change until the thing is affected by another thing, or *unstable*, where the thing will change state due to an internal transformation. A thing will keep changing state until it reaches a stable state. The transformations a thing can undergo are determined by its *transformation laws*.

Interaction is defined through the state history of a thing. If the state changes in one thing depend on the presence of another, the second is said to act on the first. Interactions may be described in terms of mutual properties and their changes. Note that while changes in mutual properties *describe* interactions, they are *not a mechanism* of such interactions.

As a thing A causes a change of state in thing B, thing B might enter an unstable state, and will being changing states due to its internal transformations. Thus, the state evolution of a thing is determined by its interactions with other things and its own transformation laws.

3 Analysis of Object Interaction

Among the principal elements of the object approach are objects, classes and communication by message passing [5, 7, 8]. UML [17] supports such communication through the following constructs: Two instances communicate by passing a stimulus, dispatched by either a 'call' or a 'send' action. A stimulus uses a *link*, which is an instance of an association, for communication. Reception of a stimulus from a 'call' action causes invocation of a *method*

whereas reception of a stimulus from a 'send' action causes a state transition. A message specifies a particular communication between instances in the context of an interaction or collaboration. We begin our examination of the involved concepts with methods.

Methods In object modeling the set of methods of an object determines the *entire* range of behavior. Ontologically, behavior of things is described through state changes, there is no construct equivalent to methods. These can be viewed as constructs rooted in software design, and thus of no ontological meaning [22]. However, object models also employ state semantics (e.g. UML state charts) which can be assigned ontological meaning by being mapped into states and state transitions [11]. Thus, in object modeling, both state semantics and methods are related to the same ontological concept of change. To interpret methods ontologically, recall that in ontology a thing stays in a stable state until another thing causes it to change. It might then enter an unstable state and keep transitioning among states. We therefore suggest the following mapping from object constructs into ontological concepts:

- A message corresponds to an event whereby one thing affects the state of another.
- A method corresponds to a transformation that must occur if the affected thing enters an unstable state.
- A state diagram describes the state evolution that occurs after the thing has entered an unstable state.

This interpretation creates an ontologically-based connection between dynamic object concepts when used for conceptual modeling which leads to the following modeling rules:

Rule 1 *The objects behavior must be completely describable by a top-level state chart.*

Rule 2 *There must be a one-to-one correspondence between the methods of an object and the transitions of the top-level state chart for that object.*

This rule allows the modeler to identify methods that may have been missed in constructing the class diagram or identify methods for which there is no corresponding state transition. The latter methods may be redundant.

According to our interpretation, the behavior modeled by a state chart describes how a thing changes states after entering an unstable state due to the action of another thing. Hence, we propose:

Rule 3 *A state chart not beginning with a state entered into as a result of an action by another object, does not provide a full description of a method.*

In the BWW-ontology states and properties are intimately linked. Accordingly, the same has been proposed for states and object attributes in UML, in particular that each state transition involves a change of at least one attribute [11]. Hence:

Rule 4 *A method must modify at least one of the set of object attributes used to define the states of the top-level state chart and at most the entire set.*

The adoption of this rule can help design and specify methods that conform to the state description.

To summarize, both methods and state charts are used for defining object behavior. Methods are implementation related concepts. Based on ontological analysis we have suggested that in conceptual modeling the concept of methods can be interpreted using ontological semantics of states transitions.

Messages In most object oriented approaches, and also in UML, interaction is *realized* through the passing of messages (stimuli) among objects. The following are typical examples of messages in object-oriented design:

- The machine sends a message to a part asking it to move itself to a new location.
- The general ledger sends a message to an office desk asking it to depreciate its value.
- A truck sends a message to the crate asking it to load itself onto the loading dock.

While such specifications are perfectly well accepted and suitable for IS design, such messages may not occur between the corresponding real world entities. [1]

[1] Note, in special circumstances, we may see message passing in the real world, e.g. between two human actors. However, this is message passing in a more special sense than message passing in the object paradigm, which is used to describe any kind of interaction.

Ontologically, interaction is defined in terms of state histories and described via changes in mutual properties. It arises because things adhere to laws that determine their allowed states and state transitions: Thing A acts-on thing B when an event in thing A changes a property which is lawfully related to a property of thing B. The BWW-Ontology does not specify *how* (i.e. the mechanism mechanism by which such interaction might be *realized*). Rather, it enables us to describe *what* interaction might occur.

It follows that while message passing is a primary concept in the object approach, and consequently in UML, it is has no direct equivalent in ontology. Rather, it is a design related concept [22, 23]. In particular, there may exist multiple ways to pass messages and also entirely different mechanisms beside messages to enact the consequences of laws.

However, using ontological analysis, we can suggest some rules to guide the use of messages in conceptual modeling to indicate interactions. In particular, the relationship between laws and interactions that satisfy them implies that messages will only be passed between objects when some of their properties are related by laws:

Rule 5 *When messages are passed between objects, properties of these objects must be related by laws.*

While there is no concept of a law in the object approach, this rule is helpful for the modeler to identify which messages might be shown in a model.

To summarize, the primary concept related to interaction independent of mechanism is laws. Message passing has no direct ontological equivalent, but can be viewed as one way to enact interactions.

Associations Associations are an element usually associated with static structure diagrams. We will show that some associations are rooted in interaction. We limit our discussion to associations in the object approach, although we realize that they are related to similar concepts in Entity-Relationship approaches [24].

There exists much debate about what associations are [e.g. 24, 9, 14]. Statements such as "relationships associate one object with another" [10] or "an association represents the relationship between objects" [1] do not further our understanding, but exemplify the con-

fusion about this construct. UML provides generalization, aggregation and composition and an unspecified ordinary association.

Some authors argue for three kinds of relationships: *Generalization*, *Aggregation* and *Use* (or message connections) [4, 7]. Since our intention here is to analyze interactions, we will not deal with generalization and aggregation here.

Rumbaugh et al. expanded the association semantics by introducing an implementation construct into conceptual modeling: "A relation expresses associations often represented in a programming language as pointers from one object to another. " [19, p. 466] "We nevertheless emphasize that associations are a useful modeling construct for .. real-world systems ..." [20, p. 31]. Martin and Odell [15] also broaden the concept arguing that associations enable the construction of conceptual networks.

Examining the sources above, we note that the ordinary association in UML encompasses two meanings. First, the 'use' association. Second, another kind argued for by Rumbaugh *et al.* [20], Martin and Odell [15] which we term *observer-dependent* as it is used to designate an observers view of the relationships between two objects. An example is a 'taller than' association in the architectural domain.

We now analyze these two types of associations ontologically. The BWW-ontology specifies three constructs that relate things: (1) Laws that relate properties of things, (2) interactions, and (3) mutual properties that can be binding or non-binding.

Binding mutual properties arise out of interaction. An example is a customer placing an order with a supplier. The notion of 'order' is the result of an interaction between the customer and the supplier. The 'order indicates that the customer and supplier interacted. Examples of non-binding mutual properties are comparative relations and spatial relations (e.g. "greater than" and "behind")[6]. We propose that these are observer-dependent as only a sentient being can impose order relations on the world. The fact that thing A is taller than thing B is a result of an imposed order, in the world there exist only things A and B with their respective heights. This concept of the BWW-ontology corresponds well with 'observer-dependent associations' as found in the object literature.

On first sight, binding mutual properties and 'use' associations may correspond as both are related to the concept of interaction. However, the semantics of 'use' associations is such

that they indicate a necessary condition for interaction to occur and must therefore exist prior to the interaction [17, p. 2-101]. This reflects the implementation related semantics of a pointer reference, without which an implementation of message passing or method calls would not be possible. Using the 'order' example: It is necessary that the customer and the supplier be linked by an association in order that the interaction 'order' (i.e. message requesting to 'order') can occur.

The situation in the ontological model is different. Ontologically, interaction occurs prior or concurrent to there being a binding mutual property: The supplier and the customer would not be linked by a binding mutual property had the 'order' event not yet occurred.

Recall, interactions in ontology are governed by laws. Hence, in ontology, the existence of a law is a prerequisite for interactions to occur (see discussion related to rule 5). As discussed above, in the object-oriented approach (and correspondingly in UML) an association is a prerequisite for interaction. We therefore propose that the closest ontological interpretation to the 'use' association is the ontological concept of *law*. Note, a law in ontology relates properties whereas an associations relates objects. Thus, we propose the following rule for the meaning of 'use' associations in conceptual modeling:

Rule 6 *A 'use' association between two objects must signify a law relating properties of the objects.*

This rule, together with rule 5, directs the modeler to identify lawfully related properties to model associations. Furthermore, it suggests, if possible, to state that law. UML provides the constructs of the Object Constraint Language OCL that can be used for this purpose. Using laws in conceptual models to describe allowed behavior can support evaluating the correctness of the final implemented system.

In summary, we note that associations in UML can be assigned one of two different ontological meanings. First, 'use' associations can be viewed as indicating the existence of laws connecting properties of different objects. Second, non 'use' associations reflect non-binding mutual properties, namely, observer-dependent relationships among properties of objects. 'Use' associations indicate possible interactions.

4 Summary and Further Research

We have examined ontologically three main concepts related to behavior and interaction in object-oriented modeling. We note that the underlying approaches to behavioral modeling are very different. Whereas UML and the object approach specify behavior through a specific mechanism, ontologically behavior is a result of the laws that govern the things. The results indicate that methods, messages and 'use' associations as originally defined have no direct ontological counterpart. This observation is in line with the suggestions of Wand [22], Wand and Weber [23] that the concepts of method and message passing are related more to the design and implementation level than to conceptual modeling. We conclude that they cannot be used directly for conceptual modeling.

However, we suggest alternative interpretations of these constructs. These interpretations lead to some modeling rules for using object concepts (and their related UML constructs) for conceptual modeling. These rules are normative and can help ensure intra- and inter-diagram consistency (e.g. between the class diagrams and state charts) As well, the rules can support the actual modeling process.

While this examination of behavioral aspects is encouraging, there remains much research to be done. In particular:

- Our results depend on the adoption of a specific ontology. Whether this is an appropriate one can only be determined empirically. The application of the rules will have to be tested by experiment in order to see if they provide useful results.

- Even if the rules seem useful in simple, experimental cases, their usefulness in practical situations will have to be tested. Such tests are quite difficult to conduct.

Thus, ultimately the final verdict on this research must be based on empirical observations.

References

[1] Bahrami, A. (1999). *Object oriented systems development.* Irwin/McGraw-Hill, Boston, MA.

[2] Bodart, F. and Weber, R. (1996). Optional properties versus subtyping in conceptual modeling: A theory and empirical test. In *Proceedings of the International Conference on Information Systems, Dec. 16-18, 1996*, page 450.

[3] Bodart, F., Sim, M., Patel, A., and Weber, R. (2001). Should optional properties be used in conceptual modelling? A theory and three empirical tests. *Information Systems Research*, **12**. Publication forthcoming.

[4] Booch, G. (1991). *Object oriented design with applications.* Benjamin/Cummings, Redwood City, CA.

[5] Booch, G. (1994). *Object Oriented Analysis and Design with Applications.* Benjamin/Cummings, Redwood City, CA.

[6] Bunge, M. A. (1977). *Ontology I: The Furniture of the World*, volume 3 of *Treatise On Basic Philosophy.* D. Reidel Publishing Company, Dordrecht, Holland.

[7] Coad, P. and Yourdon, E. (1990). *Object-Oriented Analysis.* Yourdon Press, Englewood Cliffs, NJ.

[8] Coleman, D., Arnold, P., Bodoff, S., Dollin, C., and Gilchrist, H. (1994). *Object-Oriented Development: The Fusion Method.* Prentice-Hall, Engelwood Cliffs, NJ.

[9] Dey, D., Storey, V. C., and Barron, T. M. (1999). Improving database design through the analysis of relationships. *ACM Transactions on Database Systems*, **24**(4), 453–486.

[10] Embley, D. W. (1992). *Object-oriented systems analysis: a model-driven approach.* Prentice Hall, Inc., Englewood Cliffs, NJ.

[11] Evermann, J. and Wand, Y. (2001). Towards ontologically based semantics for UML constructs. In *Proceedings of ER'01, November 2001, Yokohama.*

[12] Gemino, A. (1999). *Empirical Comparisons of Systems Analysis Modeling Techniques.* Ph.D. thesis, University of British Columbia, Canada.

[13] Green, P. and Rosemann, M. (2000). Ontological analysis of integrated process modelling. *Information Systems*, **25**(2).

[14] Kelly, S. (1995). What's in a relationship? On distinguishing property holding and object binding. In E. D. Falkenberg, W. Hesse, and A. Olive, editors, *Information System Concepts: Towards a consolidation of views.* IFIP/Chapman & Hall, London.

[15] Martin, J. and Odell, J. J. (1992). *Object-oriented analysis and design.* Prentice Hall, Englewood Cliffs, NJ.

[16] Mylopoulos, J. (1992). Conceptual modeling and telos. In P. Locuopoulos and R. Zicari, editors, *Conceptual Modeling, Databases and Cases.* John Wiley & Sons, Inc, New York et. al.

[17] OMG99 (1999). *The Unified Modelling Language Specification. Version 1.3.* OMG.

[18] Opdahl, A., Henderson-Sellers, B., and Barbier, F. (1999). An ontological evaluation of the OML metamodel. In E. Falkenberg and K. Lyytinen, editors, *Information System Concepts: An Integrated Discipline Emerging.* IFIP/Kluwer.

[19] Rumbaugh, J. (1987). Relations as semantic constructs in an object-oriented language. In *Proceedings of the 1987 Conference on Object Oriented Programming Systems and Languages and Applications, Orlando, FL.*, pages 466–481. ACM Press.

[20] Rumbaugh, J. *et al.* (1991). *Object Oriented Modeling and Design.* Prentice Hall, Englewood Cliffs, NJ.

[21] Soffer, P., Golany, B., Dori, D., and Wand, Y. (2001). Modeling off-the-shelf information systems requirements: An ontological approach. *Requirements Engineering.* Publication forthcoming.

[22] Wand, Y. (1989). A proposal for a formal model of objects. In W. Kim and F. Lchovsky, editors, *Object-oriented concepts, languages, applications and databases*, pages 537–559. ACM Press. Addison-Wesley.

[23] Wand, Y. and Weber, R. (1993). On the ontological expressiveness of information systems analysis and design grammars. *Journal of Information Systems*, (3), 217–237.

[24] Wand, Y., Storey, V. C., and Weber, R. (1999). An ontological analysis of the relationship construct in conceptual modeling. *ACM Transactions on Database Systems*, **24**(4), 494–528.